CLAIMS

What is claimed is:

1. A processor, comprising:

instruction storage in which instructions are stored;

fetch logic coupled to the instruction storage to fetch instructions from the instruction storage;

decode logic coupled to the fetch logic to decode instructions fetched by the fetch logic; and

pre-decode logic associated with the decode logic;

wherein at least some of the instructions comprise a prefix, and in parallel with the decode logic decoding a current instruction, the pre-decode logic determines whether a subsequent instruction comprise a prefix, in which case the decode logic causes a program counter to skip the prefix and changes behavior of the decode logic during decoding of the subsequent instruction.

2. The processor of claim 1, wherein at least some instructions comprise at least one Bytecode.

3. The processor of claim 1, wherein the prefix comprises a Java impdep instruction.

4 The processor of claim 3, wherein when detecting the Java impdep instruction, the subsequent instruction belongs to a different instruction set than the current instruction.

5. The processor of claim 1, wherein the prefix comprises a Java wide instruction.

6. The processor of claim 1, wherein when detecting the Java wide instruction changes format of the subsequent instruction.

7. The processor of claim 1, wherein in parallel with the decode logic decoding the current instruction, the pre-decode logic examines a predetermined number of subsequent bytes.

8. The processor of claim 7, wherein the predetermined number is at least 5.

9. A method of decoding variable length instructions, comprising:

decoding a current instruction according to a first behavior;

while decoding the current instruction, pre-decoding a subsequent instruction to determine if the subsequent instruction includes a predetermined prefix; and

if the subsequent instruction includes the predetermined prefix, causing a program counter to skip the predetermined prefix and changing the decoding of the subsequent instruction according to a second behavior.

10. The method of claim 9, wherein pre-decoding includes examining a predetermined number of bytes following the current instruction.

11. The method of claim 10, wherein the predetermined number is at least 5.

12. The method of claim 10, wherein pre-decoding further includes comparing each of the predetermined number of bytes to prefix value.

13. The method of claim 12, wherein the prefix value is equal to a Java impdep instruction.

14. The method of claim 12, wherein the prefix value is equal to a Java wide instruction.

15 The method of claim 9, wherein if the a Java wide prefix is detected, the first and second behaviors comprise a first mode for decoding instructions of a first format and a second mode for decoding instructions of a second format.

16. The method of claim 9, wherein if a Java impdep prefix is detected, the first and second behaviors comprise a first mode for decoding instructions of a first instruction set and a second mode for decoding instructions of a second instruction.

17. A system, comprising:

a main processor unit; and

a co-processor unit coupled to the main processor unit, the co-processor unit comprising:

decode logic; and

pre-decode logic associated with the decode logic;

wherein the decode logic decodes a current instruction concurrently with the pre-decode logic determining if a subsequent instruction

comprises a prefix in which case a program counter skips the prefix and changes the decode logic operation during the decoding of the subsequent instruction.

18. The system of claim 17, wherein concurrently with the decode logic decoding the current instruction, the pre-decode logic examines a predetermined number of subsequent bytes.

19. The system of claim 18, wherein the predetermined number is at least 5.

20. The system of claim 18, wherein the predetermined number of subsequent bytes is compared to a prefix value.

21. The system of claim 20, wherein the prefix value is equal to a Java wide instruction .

22. The system of claim 20, wherein the prefix value is equal to a Java impdep prefix.

23. The system of claim 17, wherein the instructions are of variable length.

24. The system of claim 17, wherein the system comprises a cellular telephone.

25. A programmable device, comprising:

a register storing a location of a current instruction;

a decode logic; and

a pre-decode logic coupled to the decode logic, wherein in parallel, the decode logic decodes the current instruction and the pre-decode logic determines if a subsequent instruction includes a prefix, and wherein if the subsequent instruction comprises the prefix, the program counter skips the prefix of the subsequent instruction and changes behavior of the decode logic for decoding of the subsequent instruction.

26. The programmable device of claim 25, wherein the prefix is a Java wide instruction.

27. The programmable device of claim 25, wherein the prefix is a Java impdep instruction.

28. The programmable device of claim 25 wherein the current instruction and the subsequent instruction each comprises at least one Bytecode.

29. The programmable device of claim 25, wherein the pre-decode logic further determines a predetermined number of subsequent bytes.

30. The programmable device of claim 29, wherein the predetermined number is at least 5.

31. The programmable device of claim 25, wherein the register is a program counter.

32. A processor, comprising:

a decode logic;

a pre-decode logic coupled to the decode logic;

means for decoding a current instruction in a first behavior in parallel with pre-decoding a subsequent instruction to determine if the subsequent instruction includes a prefix; and

means for decoding the subsequent instruction in a second behavior if the subsequent instruction includes the prefix.

33. The processor of claim 32, wherein the means for decoding a current instruction in parallel with pre-decoding a subsequent instruction further comprises pre-decoding a predetermined number of subsequent bytes.

34. The processor of claim 33, wherein the predetermined number is at least 5.